Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

X-WINDOWS SIMULATION OF STEAM POWER PLANTS BASED ON PHYSICS PRINCIPLES

Jose M. Giron-Sierra
Juan A. Gomez-Pulido
Bonifacio Andres-Toro

Departamento de Informatica y Automatica
Facultad de Fisicas
Universidad Complutense de Madrid
28040 Madrid, SPAIN

ABSTRACT

We have developed a simulation of the thermodynamic cycle of Rankine, frequently used by steam power plants. We adhered to the Object Oriented new archetype of programming, using C++. The simulation is based on a model we built from physics principles, with differential equations to consider time evolution of the variables. The modeling has been done by assigning a different object to each of the main subsystems (boiler, turbine, condenser, etc.). The simulation is coordinated by another object: the scheduler. We have two versions, for demonstrations on PCs, or SUN with X- Windows. The simulation is prepared for conceptual teaching, ready for technical refinements and the embedding of artificial intelligence.

1 INTRODUCTION

When using Object Oriented Programming (OOP), it is easy to develop a first prototype with basic functionality, useful for testing, changing, etc., in order to lay good foundations. Also, it is easy to further enrich the behavior of the objects, again test and change, etc., advancing gradually, by sure steps, to the final target. In this way, the OOP promotes an incremental procedure for developing software, with several inherent benefits (modularity, code re-use, clarity and safety). But the important decisions are needed just at the beginning: you have to think carefully about the definition of objects for the problem at hand: taxonomy, hierarchy, interactions.

Some time ago, our Department attacked the development of a training simulator for a specific steam power plant. Classic style: in FORTRAN, trying to reproduce the conditions of a control room, including a mimic panel with indicators, switches, etc. It took some years of extensive efforts. Now the power plant has been radically modified, for ecological reasons, and our training simulator needs a complete code re-writing and changes in the human interface. But we dont want to make again an ephemeral product. So we started to develop the foundations of power plants simulators, using OOP to ensure code re-use each time a plant is modified, or a new application is envisaged.

Coming to the roots, we selected the thermodynamic Rankine cycle as the reference, considering that almost every steam power plant is based on this cycle.

After four engineering years, we got a first prototype, written in C++.

At his present stage, our prototype is useful for conceptual training or engineering studies. As the workstations offer impressive graphic capabilities, with enough processing power, we decided to extend the prototype, embracing X-Windows for graphics and human interface. Compared to a mimic panel, the screens offer striking flexibility to visualize, in a motivating way, the static and dynamic data of interest for understanding purposes.

In the following, we shall describe the main aspects of our simulation.

2 STATIC CHARACTERISTICS OF THE RANKINE CYCLE

After revising a variety of reference texts (engineering thermodynamics; design, control, managing, modeling and simulation of power plants; etc.), we felt the need of developing a static simulation of the Rankine cycle, to get acquainted with the essential phenomena. The result is a program useful for conceptual study and efficiency

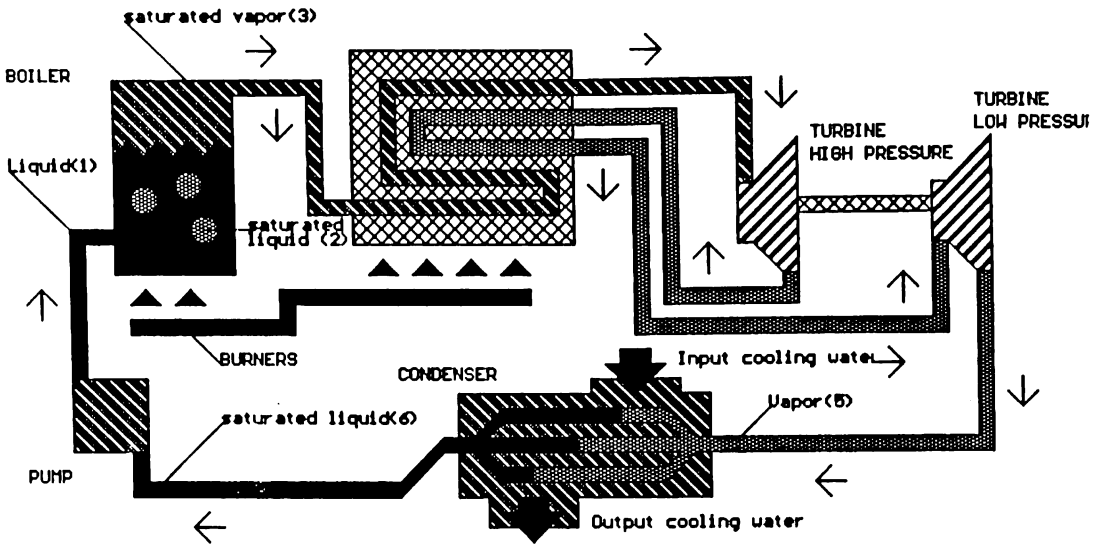

Figure 1: Screen Showing the Rankine Cycle

analysis, available for personal use on MS-DOS machines with VGA.

It is well known that water poses difficulties to the modeling of its thermodynamical behavior, as reflected by technical charts and curves. We decided to incorporate and handle the extensive empirical tables provided by the literature (Grigull et al. 1990). We established a region of interest, and arranged adequate measures to intervene when simulations go to non-realistic zones.

Our program pays attention, with texts and pictures, to some tutorial aspects. For example, one of the screens (Figure 1) shows the reheated Rankine cycle, another (Figure 2) illustrates the basic industrial structure of components to achieve the cycle, etc. For analysis purposes, the program interacts with the user, computing the characteristics of whatever realistic cycle the user specifies, and visualizing this cycle with the corresponding calculated efficiency data.

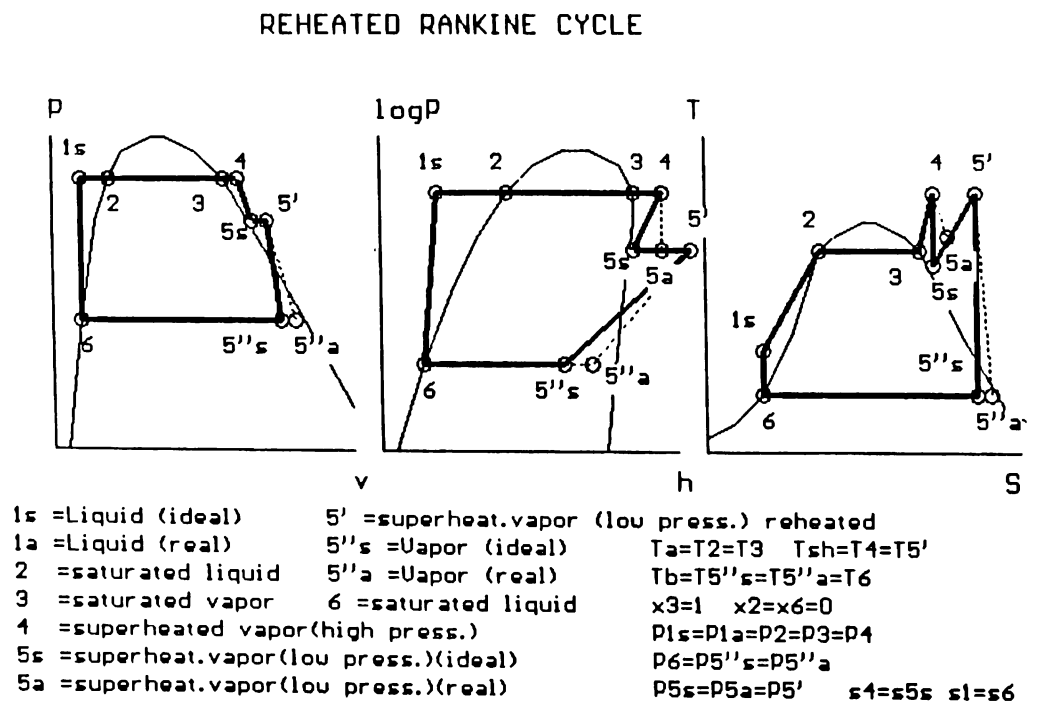

Figure 2: Screen Explainining Thermodynamics of the Cycle

3 MODELING BY COMPONENTS

Because the impact of Object Oriented technologies on software engineering, so to become the way to surmount challenges of code size, GUI integration, data safety, team work distribution and coordination, project management, amortization, etc., the general impression could be that the OOP is made for other purposes than simulation or modeling, and is difficult to adapt it for our field. But the fact is that the origin of OOP is the language SIMULA, and the name says it all: it is made for simulation. A glance to the recent literature on modeling and simulation, attest that OOP is handy for discrete-event cases, but some problems arise when considering continuous time processes. Let us comment this.

From the intuitive concept of object, it comes a natural methodology for modeling: consider as objects each of the components of a system, then interconnect the objects imitating the structure of the real system, and let each object work governed by their individual behavior and the interactions with the other objects. If you usually think in terms of block diagrams (as usual in the automatic control disciplines, and promoted now by graphical packages such SIMULINK, MATRIXx, VISSIM, etc.), you may see it almost trivial. But consider, for example, a tank with water, a pipe and a valve, so the water can leave the tank through the pipe and the valve, when you open the valve. These are three distinct objects, but the behavior of the tank is different if you open the valve, or if you close it: the behavior of an object seems to be not self-contained, but depends on other external objects.

The discussion is now open, in the field of modeling, about perspectives for assigning objects: could be components, or perhaps isolated phenomena or physical interactions, etc. If you decide to follow any alternative, then another problem comes: how to coordinate (if needed) the interactions. This is a classical topic in simulation of continuous time systems, because the real phenomena are concurrent (all the components work at the same time), but the computer is of sequential nature.

In our case, we followed the idea of assigning a different object to each of the main components of the basic industrial Rankine cycle. That means a philosophy of divide-and-conquer, to guide the modeling. Once defined a taxonomy of classes, we have to find the correct behavior of each class: the differential equations which model its dynamics. Because the handling of inheritance slow down the execution of object oriented code, we prefer to define simple hierarchies of classes, with very few levels.

4 OBJECTS FOR MODELING A BASIC POWER PLANT

By direct correspondence to the diagram of the Rankine cycle, we created the following classes:

- Burner
- Valve
- Source of Water
- Boiler
- Superheater
- Turbine
- Condenser
- Pump

Moreover, for the different states of water, we added the following classes:

- Saturated Liquid
- Steam
- Saturated Steam

Classes encapsulate protected or public variables and functions. For the building of the classes we defined the following general template:

Protected members:

State variables: dimensions and other parameters, variables related with the thermodynamical state.

Control variables: which the user can modify on simulation time.

Objects: that are embodied.

Inner functions: for internal calculations.

Public members:

Constructors: to instantiate objects with dimensions and initial values.

Calc functions: to calculate some values at a given time.

Assign functions: to assign values (calculated or returned) to several state variables.

Init functions: to calculate some values at the next interval.

Return functions: to know the values of the variables.

Control functions: to increase or decrease the values of control variables.

The set of the classes we defined, includes 154 state variables, 8 control variables, 11 constructors, 79 return functions, 16 control functions, etc.

The classes Saturated Liquid, Steam, and Saturated Steam, are very simple. For the protected section, they have only state variables (temperature, mass, entropy, enthalpy, pressure, etc.). Their public section include a constructor, one assign function to set pressure & volume (except for the Steam, that includes two assign functions for pressure & enthalphy, and pressure & entropy), and several return functions.

From the modeling perspective, the difficult task is to establish the equations describing what happens inside the components -boiler, turbine, superheater, condenser- along the closed loop. For some of the components we found enough references in the literature (Azuma 1975; Dieck-Assad 1990; Knowles 1990; Masada 1979; Usoro 1977), for some others (for instance, the condenser) we had to develop our own equations. In every case, our approach was to comply with physics principles, making a detailed analysis of mass and energy balances.

The equations of each component, are encapsulated in the related class, and, after initialization, handled through the calc functions. The Boiler includes, in the protected section, objects (instances of the Liquid and Steam classes) with information about the water contained in liquid state and in vapor state. Selecting the corresponding states, the other components of the closed loop also include the adequate objects about water. These objects are the instrument to use the tables with thermodynamics data (we took advantage from the previous work, about static characteristics, re- using the code about tables).

There are no difficulties to build the Valve, Pump, Burner, and Source of Water classes, taking into account the essential behaviors. Only the Pump requires more detailed attention.

5 SIMULATION OF DIFFERENT CONFIGURATIONS

Once created the classes needed to model the basic plant, we can instantiate objects corresponding to examples of components. For example, a boiler, a turbine, etc., having the specific dimensions of interest (when you instantiate, you set the characteristics of an object). With these objects (we shall use the term comp-objects, to recall they refer to components of a plant), we are in the position to exert several dynamic simulations, to study transients (in Khadem et al. 1990 we found some interesting orientations about compact simulators).

Because the modular nature of the comp-objects, it is possible to combine them in several ways, according to different configurations of power plants.

We created the class Scheduler, and instantiated an object SCH1. The class is designed for running various simulations, handling models, and taking care of the human and machine interfaces (menus, files). The SCH1 is an example of the flexibility provided by the class Scheduler. In SCH1 we included three different plant configurations, so it is possible to simulate any of them. Adding new configurations is a matter of inserting small, simple, pieces of code (function calls).

When running a simulation, the Scheduler send messages to the comp-objects, one after one, following the water flow along the system modeled (for example, clockwise in the case of the closed loop of Figure 3). In

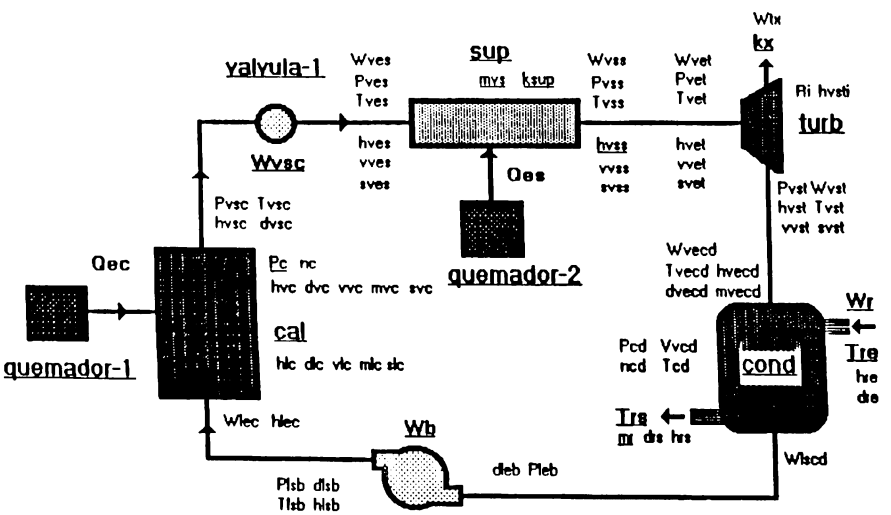

Figure 3: Objects Combined for a Basic Plant Model

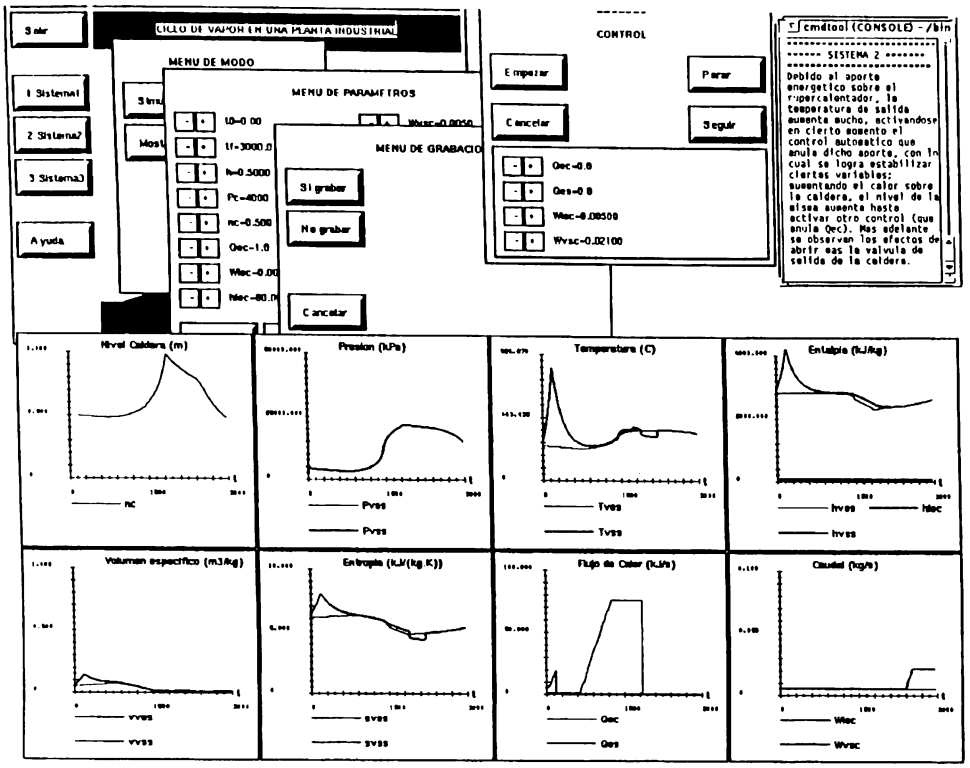

Figure 4: Planta Running a Simulation on X-Windows

response to the messages, the comp-objects change their values and send information to the Scheduler. The comp-objects with differential equations, integrate these equations to obtain the new values and change them. In this way, the Scheduler provides a solution for the problem we discussed before, of making work together the comp-objects.

6 VERSIONS. X-WINDOWS

The integrated environments offered by Borland languages (Borland 1990) are really beneficial for a rapid development of tested code. So we used Borland C++ to build a first version of an application we call **Planta.** This application include the water thermodynamics tables, the comp-objects, and SCH1. The human interface is solved using BGI graphics.

Having approved the initial basis, we developed a second version, more complete, for a SUN Sparcstation and GNU G++ (public domain C++). We had a gratifying experience of excellent portability from the MS-DOS machine and Borland code, to the new platform.

The world of workstations offer X-Windows as the important standard for a friendly human interface, on the basis of a protocol for communication between different terminals and computers. With X-Windows you have a flexible GUI with windows and mouse, being supported by distinct computing structures: one workstation alone, X-terminals and host computers, local networks: the fact is that X-Windows id designed for hardware-independent distributed computing. There is now abundant literature about X-Windows, for users, system administrators, programmers (we employed Barkakati 1991), etc.

Because the evident advantages of X-Windows, we determined to use it for our application. SCH1 is the object which includes the functions to handle X-Windows as the means for mouse & windows, and for graphics.

Several manners of programming are possible with X-Windows, because it includes a variety of function libraries you can use. These libraries are built following a hierarchy: at the low level Xlib provides basic functions, that are employed by the next levels up (Xt intrinsics, X Widget Set, among other toolkits and complements), to offer some structural units you can exploit for artistic menus and nice operating metaphors. Usually, the more beautiful, the more slow.

We were told that OOP could produce slow applications. Actually, our previous experience with SMALLTALK was that inheritance mechanisms take precious time, so is better to define simple hierarchies. As we wanted to evaluate how fast our simulations can run (calculations, data processing), we decided to employ Xlib to have the less delay for screen graphics.

Respect with the human interface, the main duty of SCH1 is handling events through the event-driven X programming model. Xlib furnish its context manager utility routines for this purpose.

To compare the impact of hardware on the speed, we run the same simulation experiment with the MS-DOS and with the X-Windows versions of **Planta.** Here are the results:

MS-DOS Planta,

on 80386, 25 Mz, no copro.: 45 minutes, 3 seconds

on 80386, 33 Mz, with copro: 9 minutes

on 80486, 33 Mz : 4 minutes 15 seconds

X-Windows Planta,

on Sun Sparc 1+ : 37 seconds

The big screen of the Sun Workstation permits to show a profuse set of windows, with curves visualizing the dynamical behavior of the thermodynamical magnitudes of interest, while running the simulation of a plant (see Figure 4). We have speed enough to increase the contents of the objects, and the structural difficulty of the examples.

CONCLUSION

We wanted to know if OOP could be used for Steam Power Plants simulations: the advantages are clear, but there are some preventions about speed and connectivity of comp-objects. With the development of **Planta,** we got evidence that it is possible to obtain Object Oriented combined models and fast simulations for our case.

The application **Planta** is written in C++, use X-Windows, and is portable to any Workstation. The listing of **Planta** takes 250 K of text. **Planta** is interactive: during the simulations, the user can change some parameters (for example, closing a valve, or increasing heat, etc.) and see the effects on the process evolution. With **Planta** we have a basis for the simulation of different examples of Steam Power Plants.

In this moment we have two versions of **Planta**, for MS-DOS and for Sun Sparcstation, ready for demonstration. We are writing extensive documentation about the models (deriving the equations), and the Planta users manual.

Next steps of our research will be the embedding of artificial intelligence, by making each object a mini-expert with a specific set of rules, and the development of a graphic editor to build examples of plants.

REFERENCES

Azuma, A. 1975. Modeling and Simulation of a Steam Power Station. M.I.T. Thesis.

Barkakati, N. 1991. X-Windows System Programming. SAMS Mac Millan.

Borland Intl. 1990. Turbo C++ Users and Programmers Guide.

Dieck-Assad, G. 1990. Development of a State Space Boiler Model for Process Optimization. *Simulation*, October 1990.

Grigull, U. et al. 1990. Steam Tables in SI-Units. Springer Verlag.

Khadem, M. et al. 1986. A Compact, Interactive and Color-Graphics Based Simulator for Power Plant Analysis. *6th. Power Plant Dynamics, Control & Testing Symposium.*

Knowles, J.B. 1990. Simulation and Control of Electrical Power Stations. John Wiley.

Masada, G. 1979. Modeling and Control of Power Plant Boiler-Turbine- Generator Systems. M.I.T. Thesis.

Usoro, P.B. 1977. Modeling and Simulation of a Drum Boiler-Turbine Power Plant Under Emergency State Control. M.I.T. Thesis.

AUTHOR BIOGRAPHIES

JOSE M. GIRON-SIERRA is a Professor in the Department of Computer Science and Automatic Control at the University Complutense of Madrid, Spain, where he received the Ph.D. degree in 1978. His research interests are the simulation of chemical, energy, and manufacturing industrial processes, with emphasis on the integration of modern technologies. He is a member of the board organizing the Spanish Simulation Society.

JUAN A. GOMEZ-PULIDO is an Assistant Professor in the Department of Computer Science and Automatic Control at the University Complutense of Madrid, Spain, where he received the Ph.D. degree in 1993. His research interests are the simulation of energy industrial processes, integrating modern technologies.

BONIFACIO ANDRES-TORO is an Associate Professor in the Department of Computer Science and Automatic Control at the University Complutense of Madrid, Spain. His research interests are the simulation of chemical industrial processes, integrating artificial intelligence technologies.